Solving PDEs on Overlapping Grids with Overture

Bill Henshaw

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA

University of Illinois at Urbana-Champaign

ILLINOIS

September 2010.

Acknowledgments.

Supported by

Department of Energy, Office of Science

ASCR Applied Math Program

LLNL: Laboratory Directed Research and Development (LDRD) program

Current Overture developers

Kyle Chand

Bill Henshaw

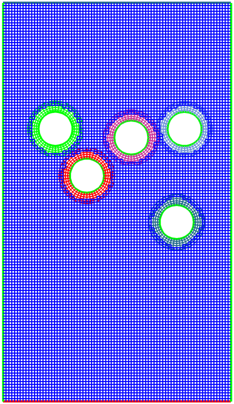
Major Contributors

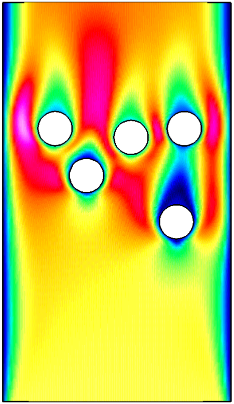
Don Schwendeman (RPI),

Jeff Banks (LLNL).

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

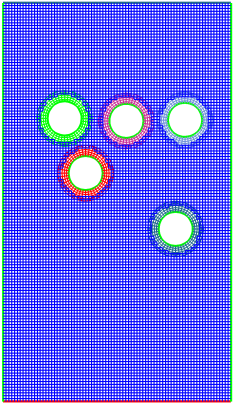


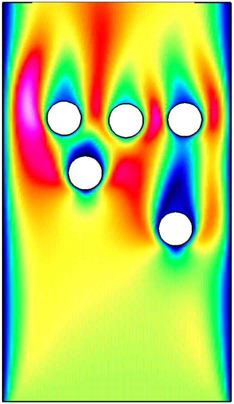


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Efficient for high-order accurate methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

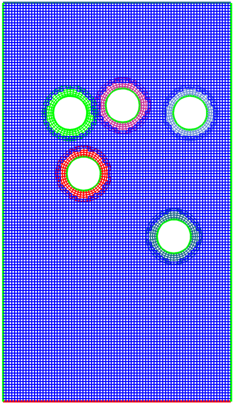


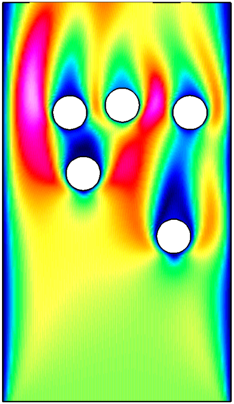


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Efficient for high-order accurate methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

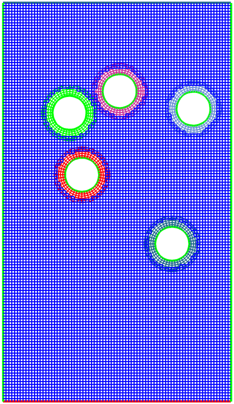


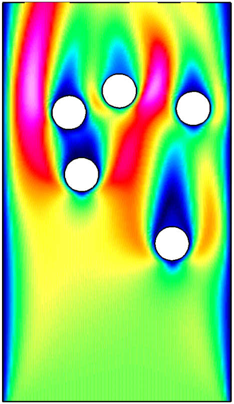


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Efficient for high-order accurate methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

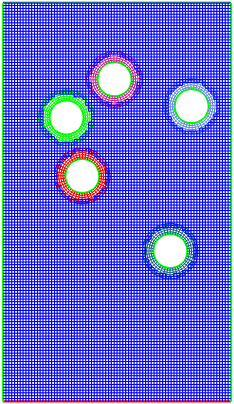


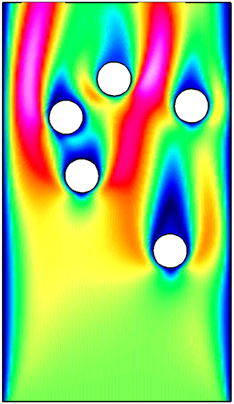


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Efficient for high-order accurate methods.

What are overlapping grids and why are they useful?

Overlapping grid: a set of structured grids that overlap.

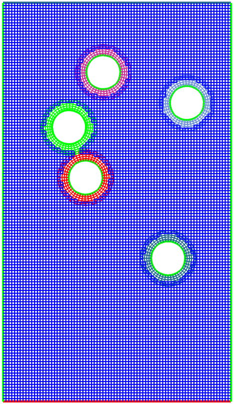


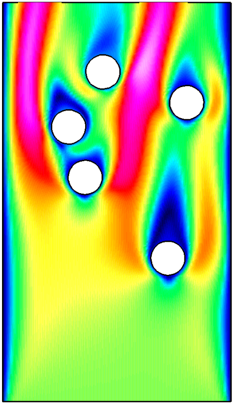


- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Efficient for high-order accurate methods.

What are overlapping grids and why are they useful?

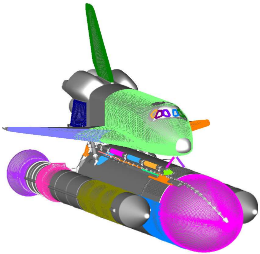
Overlapping grid: a set of structured grids that overlap.

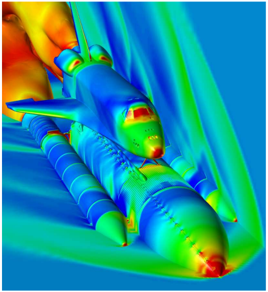


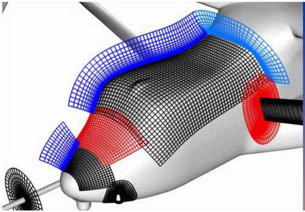


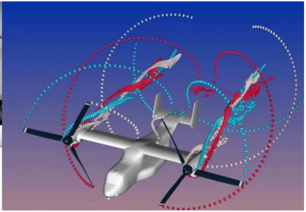
- Overlapping grids can be rapidly generated as bodies move.
- High quality grids under large displacements.
- Cartesian grids for efficiency.
- Efficient for high-order accurate methods.

Aerospace applications using overlapping grids.









Space shuttle figures courtesy of William Chan and Reynaldo Gomez.
V-22 Osprey figures courtesy of William Chan, Andrew Wissink and Robert Meakin.

Overture: tools for solving PDE's on overlapping grids

- high level C++ interface for rapid prototyping of PDE solvers.
- built upon optimized C and fortran kernels.
- library of finite-difference operators: conservative and non-conservative, 2nd, 4th, 6th and 8th order accurate approximations.
- support for moving grids.
- support for block structured adaptive mesh refinement (AMR).
- extensive grid generation capabilities (Ogen).
- CAD fixup tools (for CAD from IGES files).
- interactive graphics and data base support (HDF).

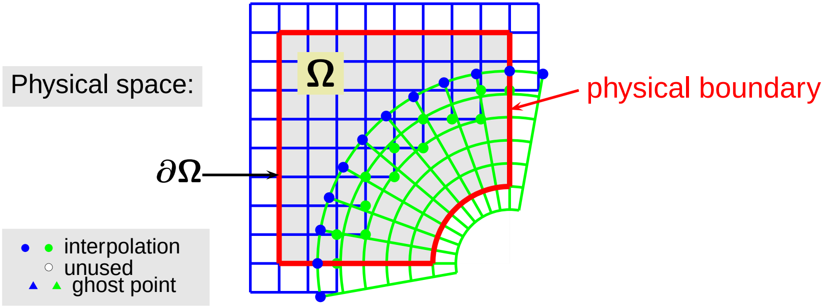
The CG (Composite Grid) suite of PDE solvers

- cgad: advection diffusion equations.
- cgins: incompressible Navier-Stokes with heat transfer.
- cgcns: compressible Navier-Stokes, reactive Euler equations.
- cgmp: multi-physics solver (e.g. conjugate heat transfer).
- cgmx: time domain Maxwell's equations solver.
- cgsm: solid mechanics (*new*)

Overture and CG are freely available from the web:

www.llnl.gov/CASC/Overture

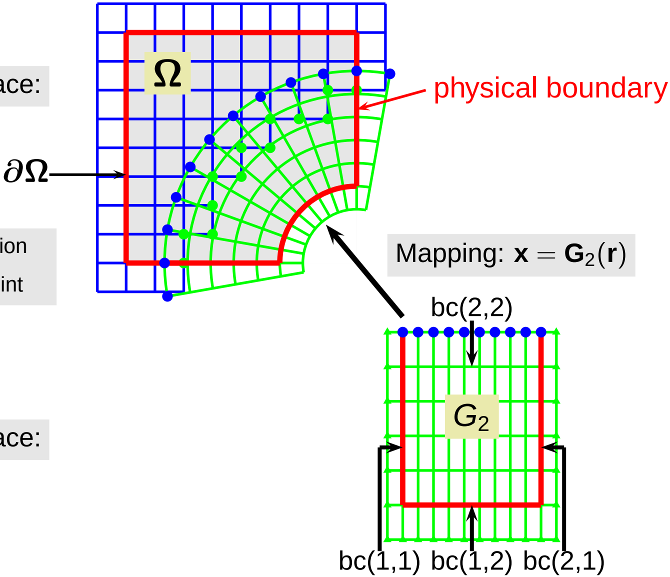
Components of an Overlapping Grid



Components of an Overlapping Grid

Physical space:

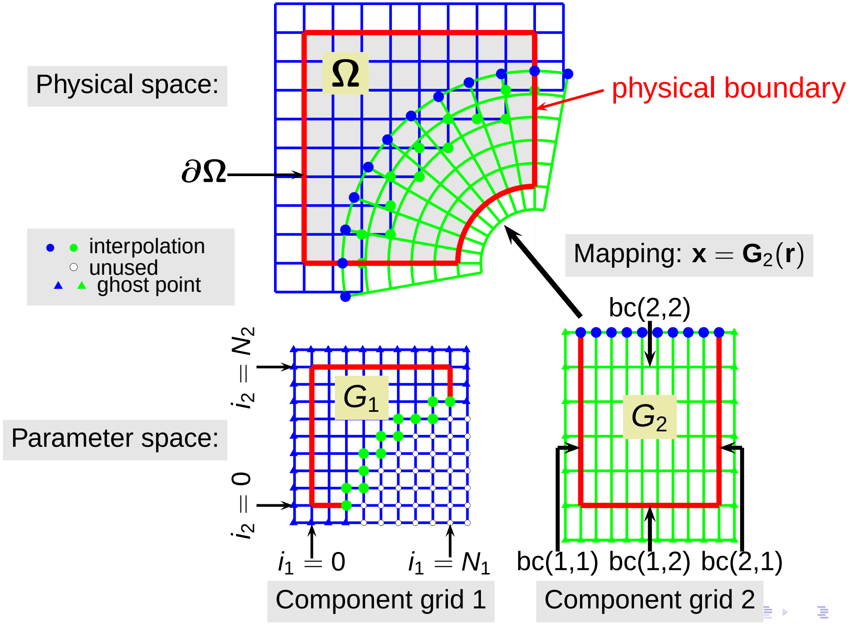
- interpolation
- unused
- ghost point



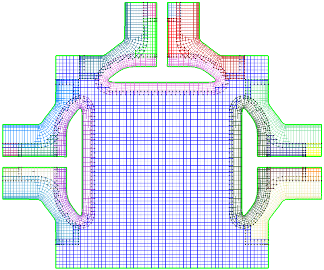
Parameter space:

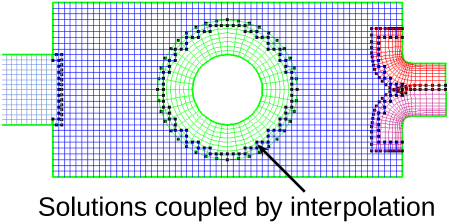
Component grid 2

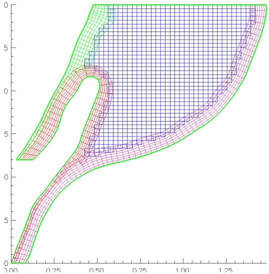
Components of an Overlapping Grid



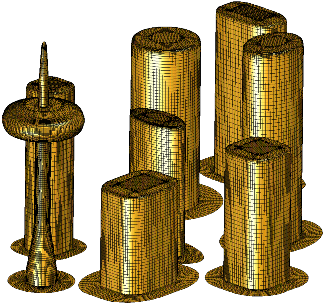
Ogen can be used to build 2D overlapping grids:

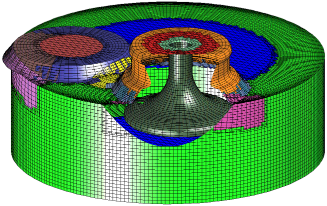


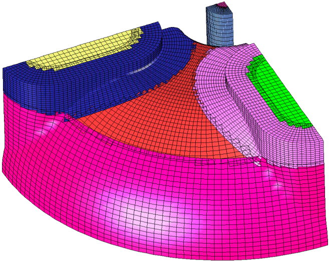


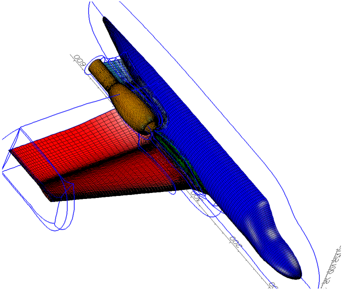


Ogen can be used to build 3D overlapping grids:









Composite/ Chimera/ Overset/ Overlapping Grids

A Short History

- Volkov, circa [1966] developed a *Composite Mesh* method for Laplace's equation on regions with piece-wise smooth boundaries separated by corners. Polar grids are fitted at corners to handle potential singularities.
- Starius, circa [1977] (student of H.-O. Kreiss) considered *Composite Mesh* methods for elliptic and hyperbolic problems – introduces a hyperbolic grid generator.
- Steger, circa [1980] independently conceives the idea of the overlapping grid, subsequently named the *Chimera* approach after the mythical Chimera beast having a human face, a lion's mane and legs, a goat's body, and dragon's tail. NASA groups develop grid generator PEGSUS, hyperbolic grid generation and flow solver Overflow (Steger, Benek, Suhs, Buning, Chan, Meakin, et. al.)
- B. Kreiss circa [1980] develops overlapping grid generator which subsequently leads to the CMPGRD grid generator [1983] (Chesshire, Henshaw) later leading to the Overture set of tools [1994].

- Volkov, circa [1966] developed a *Composite Mesh* method for Laplace's equation on regions with piece-wise smooth boundaries separated by corners. Polar grids are fitted at corners to handle potential singularities.
- Starius, circa [1977] (student of H.-O. Kreiss) considered *Composite Mesh* methods for elliptic and hyperbolic problems – introduces a hyperbolic grid generator.
- Steger, circa [1980] independently conceives the idea of the overlapping grid, subsequently named the *Chimera* approach after the mythical Chimera beast having a human face, a lion's mane and legs, a goat's body, and dragon's tail. NASA groups develop grid generator PEGSUS, hyperbolic grid generation and flow solver Overflow (Steger, Benek, Suhs, Buning, Chan, Meakin, et. al.)
- B. Kreiss circa [1980] develops overlapping grid generator which subsequently leads to the CMPGRD grid generator [1983] (Chesshire, Henshaw) later leading to the Overture set of tools [1994].

Composite/ Chimera/ Overset/ Overlapping Grids

A Short History

- Volkov, circa [1966] developed a *Composite Mesh* method for Laplace's equation on regions with piece-wise smooth boundaries separated by corners. Polar grids are fitted at corners to handle potential singularities.
- Starius, circa [1977] (student of H.-O. Kreiss) considered *Composite Mesh* methods for elliptic and hyperbolic problems – introduces a hyperbolic grid generator.
- Steger, circa [1980] independently conceives the idea of the overlapping grid, subsequently named the *Chimera* approach after the mythical Chimera beast having a human face, a lion's mane and legs, a goat's body, and dragon's tail. NASA groups develop grid generator PEGSUS, hyperbolic grid generation and flow solver Overflow (Steger, Benek, Suhs, Buning, Chan, Meakin, et. al.)
- B. Kreiss circa [1980] develops overlapping grid generator which subsequently leads to the CMPGRD grid generator [1983] (Chesshire, Henshaw) later leading to the Overture set of tools [1994].

- Volkov, circa [1966] developed a *Composite Mesh* method for Laplace's equation on regions with piece-wise smooth boundaries separated by corners. Polar grids are fitted at corners to handle potential singularities.
- Starius, circa [1977] (student of H.-O. Kreiss) considered *Composite Mesh* methods for elliptic and hyperbolic problems – introduces a hyperbolic grid generator.
- Steger, circa [1980] independently conceives the idea of the overlapping grid, subsequently named the *Chimera* approach after the mythical Chimera beast having a human face, a lion's mane and legs, a goat's body, and dragon's tail. NASA groups develop grid generator PEGSUS, hyperbolic grid generation and flow solver Overflow (Steger, Benek, Suhs, Buning, Chan, Meakin, et. al.)
- B. Kreiss circa [1980] develops overlapping grid generator which subsequently leads to the CMPGRD grid generator [1983] (Chesshire, Henshaw) later leading to the Overture set of tools [1994].

Theory for finite difference schemes

There is extensive numerical analysis theory underpinning this work.

- classic von Neumann stability analysis (periodic domains).
- energy estimates (L_2-norm estimates).
- normal mode analysis, GKS theory (initial boundary value problems).

Some references:

- Gustafsson, Kreiss, Oliger, *Time Dependent Problems and Difference Methods*, (book).
- Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, (book).
- Gustafsson, Kreiss, Sundström, *Stability Theory of Difference Approx. for Mixed Initial Boundary Value Problems, I. and II.*, Math. Comp.
- Starius, *On Composite Mesh Difference Methods for Hyperbolic Differential Equations*, Numer. Math.

A one-dimensional overlapping grid example

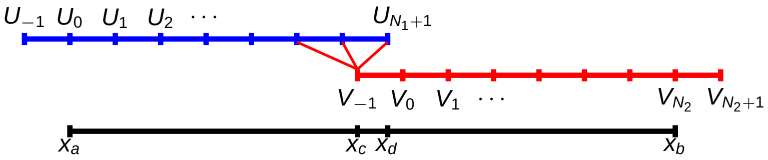
To solve the advection-diffusion equation

$$\begin{aligned} u_t + a u_x &= \nu u_{xx}, && x \in (0,1) \\ u(0,t) = g_0(t), \quad u_x(1,t) &= g_1(t), && \text{(boundary conditions)} \\ u(x,0) &= u_0(x), && \text{(initial conditions)} \end{aligned}$$

introduce grid points on the two overlapping component grids,

$$\begin{aligned} x_i^{(1)} &= x_a + i\Delta x_1, & i &= -1,0,1,\ldots,N_1+1, & \Delta x_1 &= (x_d - x_a)/N_1 \\ x_j^{(2)} &= x_c + (j+1)\Delta x_2, & j &= -1,0,1,\ldots,N_2+1, & \Delta x_2 &= (x_b - x_c)/N_2 \end{aligned}$$

and approximations $U_i^n \approx u(x_i^{(1)}, n\Delta t)$, $V_i^n \approx u(x_i^{(2)}, n\Delta t)$.



Discretize with forward-Euler and central differences

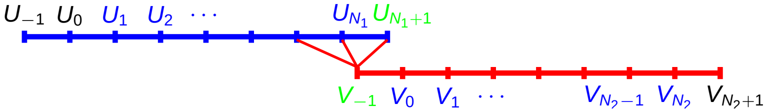
Given the solution at time t^n, compute the solution at time t^{n+1}:

$$
\begin{aligned}
(U_i^{n+1} - U_i^n)/\Delta t &= -aD_0 U_i^n + \nu D_+ D_- U_i^n, \qquad i = 1, 2, \ldots, N_1 \\
(V_j^{n+1} - V_j^n)/\Delta t &= -aD_0 V_j^n + \nu D_+ D_- V_j^n, \qquad j = 0, 2, \ldots, N_2 \\
U_0^{n+1} &= g(t^n), \quad D_0 V_{N_2}^{n+1} = g_1(t^{n+1}), \qquad \text{(boundary conditions)}
\end{aligned}
$$

$$
U_{N_1+1}^{n+1} = (1-\alpha)(1-\frac{\alpha}{2})\, V_{-1}^{n+1} + \alpha(2-\alpha)\, V_0^{n+1} + \frac{\alpha}{2}(\alpha - 1)\, V_1^{n+1}, \quad \text{(interpolation)}
$$

$$
V_{-1}^{n+1} = (1-\beta)(1-\frac{\beta}{2})\, U_{N_1-1}^{n+1} + \beta(2-\beta)\, U_{N_1}^{n+1} + \frac{\beta}{2}(\beta - 1)\, U_{N_1+1}^{n+1}, \quad \text{(interpolation)}
$$

$$
D_0 U_i^n = \frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x}, \quad D_+ U_i^n = \frac{U_{i+1}^n - U_i^n}{\Delta x}, \quad D_- U_i^n = \frac{U_i^n - U_{i-1}^n}{\Delta x}.
$$



Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg;     // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg);     // create a grid function
u=1.;
CompositeGridOperators op(cg);    // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
   u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) );     // forward Euler
   t+=dt;
   u.interpolate();
   u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
   u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg;     // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg);     // create a grid function
u=1.;
CompositeGridOperators op(cg);     // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
   u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) );     // forward Euler
   t+=dt;
   u.interpolate();
   u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
   u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg;     // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg);     // create a grid function
u=1.;
CompositeGridOperators op(cg);     // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
  u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) );     // forward Euler
  t+=dt;
  u.interpolate();
  u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
  u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

```
CompositeGrid cg;      // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg);     // create a grid function
u=1.;
CompositeGridOperators op(cg);    // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
   u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) );     // forward Euler
   t+=dt;
   u.interpolate();
   u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
   u.finishBoundaryConditions();
}
```

Overture supports a high-level C++ interface

But is built upon mainly Fortran kernels.

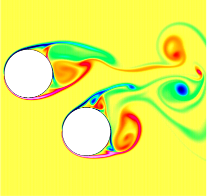
Solve $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$

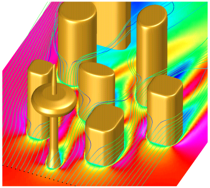
```
CompositeGrid cg;      // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg);      // create a grid function
u=1.;
CompositeGridOperators op(cg);     // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
   u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) );     // forward Euler
   t+=dt;
   u.interpolate();
   u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
   u.finishBoundaryConditions();
}
```

Overture is used by research groups worldwide

- Blood flow in veins with blood clot filters. (Mike Singer, LLNL).
- Pitching airfoils and micro-air vehicles (Yongsheng Lian, U. of Louisville)
- Relativistic hydrodynamics and Einstein field equations (Philip Blakely, Nikos Nikiforakis, U. Cambridge).
- Compressible flow/ice-formation (Graeme Leese, U. Cambridge).
- Tear films and droplets (Rich Braun U. Delaware, Kara Maki UMN).
- High-order accurate subsonic/transonic aero-acoustics (Phillipe Lafon, CNRS, EDF, France).
- Low Reynolds flow for pitching airfoils (D. Chandar, R. Yapalparvi, M. Damodaran, NTU, Singapore).
- Incompressible flow in pumps (J.P. Potanza, Shell Oil, Houston).
- High-order accurate, compact Hermite-Taylor schemes (Tom Hagstrom, SMU, Dallas).

Cgins: incompressible Navier-Stokes solver.





- 2nd-order and 4th-order accurate (DNS).
- support for moving rigid-bodies (not parallel yet).
- heat transfer (Boussinesq approximation).
- semi-implicit (time accurate), pseudo steady-state (efficient line solver), full implicit.

• WDH., *A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids*, J. Comput. Phys, **113**, no. 1, (1994) 13–25.

Incompressible Navier-Stokes.

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} = 0, \qquad t>0, \quad \mathbf{x}\in\Omega$$
$$\nabla\cdot\mathbf{u} = 0 \qquad t>0, \quad \mathbf{x}\in\Omega$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u}=0, \text{ (pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,).$$

Use $\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

- WDH, N.A. Petersson, *A Split-Step Scheme for the Incompressible Navier-Stokes Equations*, 2003.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega \\ \Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega \end{aligned}$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \ \text{(pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,).$$

Use $\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

- WDH, N.A. Petersson, *A Split-Step Scheme for the Incompressible Navier-Stokes Equations*, 2003.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$
\begin{aligned}
\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega \\
\Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega
\end{aligned}
$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$
\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \ \text{(pressure BC)} \quad \mathbf{x}\in\partial\Omega,
$$

with *numerical boundary condition*:

$$
p_n = -\mathbf{n}\cdot(\nu\nabla\times\nabla\times\mathbf{u}).
$$

Use $\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

- WDH, N.A. Petersson, *A Split-Step Scheme for the Incompressible Navier-Stokes Equations*, 2003.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega \\ \Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega \end{aligned}$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \ \text{(pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,).$$

Use $\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

- WDH, N.A. Petersson, *A Split-Step Scheme for the Incompressible Navier-Stokes Equations*, 2003.

Incompressible Navier-Stokes.

Split-step, velocity-pressure formulation:

$$\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega \\ \Delta p - \nabla\mathbf{u} : \nabla\mathbf{u} - \alpha\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} &= 0, \qquad t>0, \quad \mathbf{x}\in\Omega \end{aligned}$$

Divergence damping term: $\alpha\nabla\cdot\mathbf{u}$ is important.

Wall boundary conditions:

$$\mathbf{u} = 0, \quad \nabla\cdot\mathbf{u} = 0, \ \text{(pressure BC)} \quad \mathbf{x}\in\partial\Omega,$$

with *numerical boundary condition*:

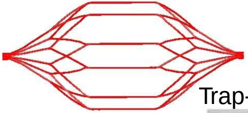
$$p_n = -\mathbf{n}\cdot(\,\nu\nabla\times\nabla\times\mathbf{u}\,).$$

Use $\nabla\times\nabla\times\mathbf{u}$ instead of $\Delta\mathbf{u}$ for implicit time-stepping.

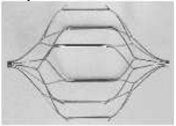
- WDH, N.A. Petersson, *A Split-Step Scheme for the Incompressible Navier-Stokes Equations*, 2003.

Flow past a blood-clot filter using cgins

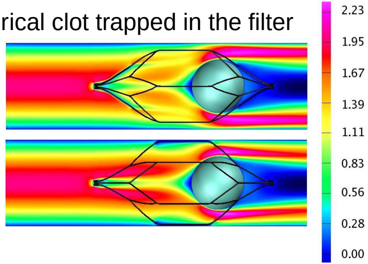
Overlapping grid for the filter



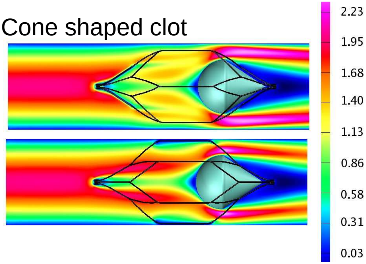
Trap-ease wire filter



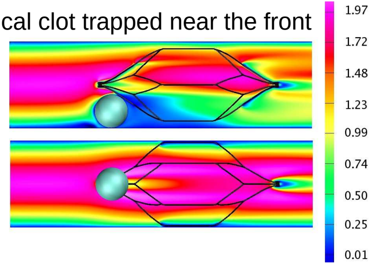
Spherical clot trapped in the filter



Cone shaped clot

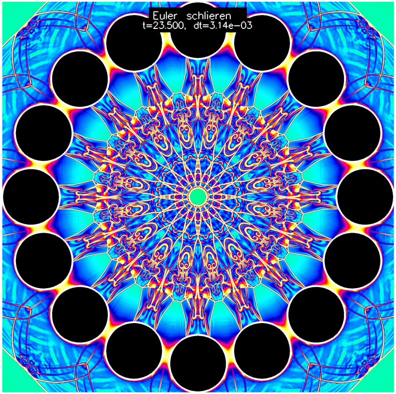


Spherical clot trapped near the front



M.A. Singer, WDH, S.L. Wang, *Computational Modeling of Blood Flow in the Trapease Inferior Vena Cava Filter*, Journal of Vascular and Interventional Radiology, **20**, 2009.

Cgcns: compressible N-S and reactive-Euler.

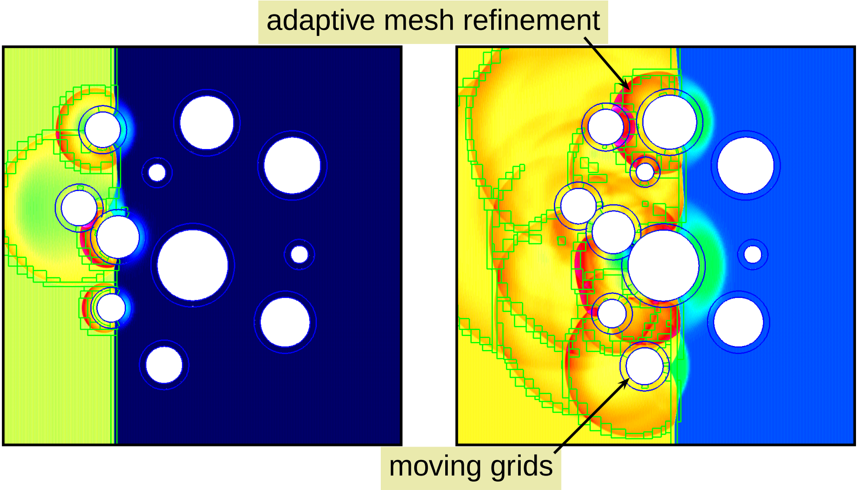


- reactive and non-reactive Euler equations, Don Schwendeman (RPI).
- compressible Navier-Stokes.
- multi-fluid formulation, Jeff Banks (LLNL).
- adaptive mesh refinement and moving grids.

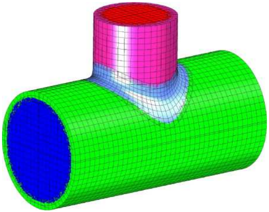
• WDH., D. W. Schwendeman, *Parallel Computation of Three-Dimensional Flows using Overlapping Grids with Adaptive Mesh Refinement*, J. Comp. Phys. **227** (2008).
• WDH., DWS, *Moving Overlapping Grids with Adaptive Mesh Refinement for High-Speed Reactive and Nonreactive Flow*, J. Comp. Phys. **216** (2005).
• WDH., DWS, An adaptive numerical scheme for high-speed reactive flow on overlapping grids, J. Comp. Phys. **191** (2003).

Moving overlapping grids and AMR

A shock hitting a collection of cylinders (density).

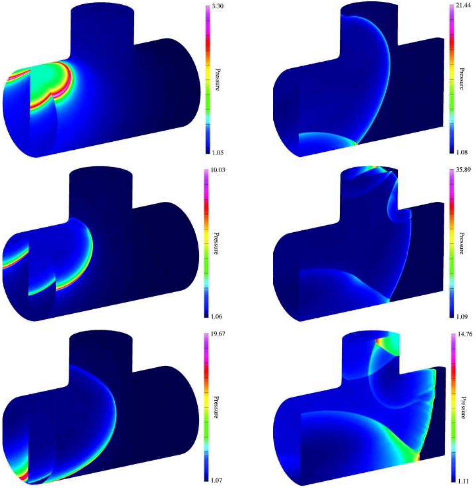


Detonation initiation in a T-shaped pipe



overlapping grid

pressure



Notes: cgcns, reactive-Euler: one refinement level, factor 4, 4930 time steps, 48 processors, from 5 to 682 grids, 100M pts (max) (eff. resolution 400 M).

Estimating Convergence Rates

Define the volume-weighted discrete L_p-norm of a grid function $U_{\mathbf{i}}$ as

$$\|U_{\mathbf{i}}\|_p = \left(\frac{\sum_{\mathbf{i}} |U_{\mathbf{i}}|^p \, d\mathcal{V}_{\mathbf{i}}}{\sum_{\mathbf{i}} d\mathcal{V}_{\mathbf{i}}}\right)^{1/p}, \qquad d\mathcal{V}_{\mathbf{i}} = \left|\frac{\partial \mathbf{x}}{\partial \mathbf{r}}\right|_{\mathbf{i}} dr_1 dr_2 dr_3.$$

Assume the discrete solution $U_{\mathbf{i}}^m$ at grid spacing h_m satisfies

$$U_{\mathbf{i}}^m - u(\mathbf{x}_{\mathbf{i}}^m, t) \approx C_{\mathbf{i}}^m h_m^\mu,$$

The difference between resolution h_n and h_m is

$$\|U_{\mathbf{i}}^m - \mathcal{R}_n^m U_{\mathbf{i}}^n\|_p \approx C|h_m^\mu - h_n^\mu|,$$

where $\mathcal{R}_n^m$ is a fine to coarse restriction operator.

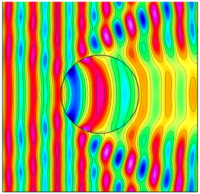
Result: Given three solutions we can estimate the convergence rate μ and the error.

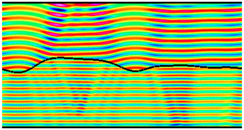
Estimating Convergence Rates

Detonation in a T-Pipe				
	$t = 2.0$		$t = 2.8$	
h_m	$\mathcal{E}_1^m$	$\mathcal{E}_2^m$	$\mathcal{E}_1^m$	$\mathcal{E}_2^m$
$1/120$	4.0e$-$3	3.0e$-$2	3.8e$-$2	2.6e$-$1
$1/160$	2.2e$-$3	1.6e$-$2	2.4e$-$2	1.9e$-$1
$1/240$	9.8e$-$4	7.1e$-$3	1.2e$-$2	1.2e$-$1
rate, μ	2.04	2.07	1.65	1.09

Estimated L_1 and L_2 errors in the density, $\mathcal{E}_1^m$ and $\mathcal{E}_2^m$, respectively, and convergence rates μ at $t = 2.0$ and $t = 2.8$.

Cgmx: electromagnetics solver.





- a time-domain finite difference scheme.
- fourth-order accurate, 2D, 3D.
- Efficient time-stepping with the modified-equation approach
- High-order accurate symmetric difference approximations.
- High-order-accurate *centered* boundary and interface conditions.

• WDH., *A High-Order Accurate Parallel Solver for Maxwell's Equations on Overlapping Grids*, SIAM J. Scientific Computing, **28**, no. 5, (2006).

Maxwell's equations are solved in second-order form

Maxwell's equations:

$$\epsilon\mu\ \partial_t^2\mathbf{E} = \Delta\mathbf{E} + \nabla\Big(\nabla \ln\epsilon\ \cdot \mathbf{E}\Big) + \nabla\ln\mu \times \Big(\nabla\times\mathbf{E}\Big) - \mu\partial_t\mathbf{j}$$

$$\epsilon\mu\ \partial_t^2\mathbf{H} = \Delta\mathbf{H} + \nabla\Big(\nabla \ln\mu\ \cdot \mathbf{H}\Big) + \nabla\ln\epsilon \times \Big(\nabla\times\mathbf{H}\Big) + \epsilon\nabla\times(\frac{1}{\epsilon}\mathbf{j})$$

Advantages of the second-order form:

- No need for a staggered grid since the operator Δ is elliptic.
- One can solve for **E** alone.

Maxwell's equations are solved in second-order form

Maxwell's equations:

$$\epsilon\mu\ \partial_t^2 \mathbf{E} = \Delta \mathbf{E} + \nabla\Big(\nabla \ln \epsilon\ \cdot \mathbf{E}\Big) + \nabla \ln \mu \times \Big(\nabla \times \mathbf{E}\Big) - \mu \partial_t \mathbf{j}$$

$$\epsilon\mu\ \partial_t^2 \mathbf{H} = \Delta \mathbf{H} + \nabla\Big(\nabla \ln \mu\ \cdot \mathbf{H}\Big) + \nabla \ln \epsilon \times \Big(\nabla \times \mathbf{H}\Big) + \epsilon \nabla \times (\frac{1}{\epsilon}\mathbf{j})$$

Advantages of the second-order form:

- No need for a staggered grid since the operator Δ is elliptic.
- One can solve for $\mathbf{E}$ alone.

Modified Equation time stepping

Taylor series in time:

$$\frac{u(t+\Delta t)-2u(t)+u(t-\Delta t)}{\Delta t^2} = u_{tt} + \frac{\Delta t^2}{12} u_{tttt} + O(\Delta t^4)$$

For the wave equation

$$u_{tt} = \Delta u$$

a fourth-order scheme in space and time is

$$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{\Delta t^2} = \Delta_{4h} U_i^n + \frac{\Delta t^2}{12} (\Delta^2)_{2h} U_i^n$$

This scheme is very efficient (especially on Cartesian grids) and allows a large (cfl=1) time step.

Modified Equation time stepping

Taylor series in time:

$$\frac{u(t+\Delta t) - 2u(t) + u(t-\Delta t)}{\Delta t^2} = u_{tt} + \frac{\Delta t^2}{12} u_{tttt} + O(\Delta t^4)$$

For the wave equation

$$u_{tt} = \Delta u$$

a fourth-order scheme in space and time is

$$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{\Delta t^2} = \Delta_{4h} U_i^n + \frac{\Delta t^2}{12} (\Delta^2)_{2h} U_i^n$$

This scheme is very efficient (especially on Cartesian grids) and allows a large (cfl=1) time step.

Modified Equation time stepping

Taylor series in time:

$$\frac{u(t+\Delta t) - 2u(t) + u(t-\Delta t)}{\Delta t^2} = u_{tt} + \frac{\Delta t^2}{12} u_{tttt} + O(\Delta t^4)$$

For the wave equation

$$u_{tt} = \Delta u$$

a fourth-order scheme in space and time is

$$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{\Delta t^2} = \Delta_{4h} U_i^n + \frac{\Delta t^2}{12} (\Delta^2)_{2h} U_i^n$$

This scheme is very efficient (especially on Cartesian grids) and allows a large (cfl=1) time step.

Modified Equation time stepping

Taylor series in time:

$$\frac{u(t+\Delta t) - 2u(t) + u(t-\Delta t)}{\Delta t^2} = u_{tt} + \frac{\Delta t^2}{12} u_{tttt} + O(\Delta t^4)$$

For the wave equation

$$u_{tt} = \Delta u$$

a fourth-order scheme in space and time is

$$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{\Delta t^2} = \Delta_{4h} U_i^n + \frac{\Delta t^2}{12} (\Delta^2)_{2h} U_i^n$$

This scheme is very efficient (especially on Cartesian grids) and allows a large (cfl=1) time step.

Centered numerical boundary conditions for high-order approximations

Vector wave equation on a square

$$\mathbf{E}_{tt} = \mathbf{E}_{xx} + \mathbf{E}_{yy} \quad \mathbf{x} \in \Omega = [0,1]^2$$

PEC (perfect electrical conductor) boundary at $x = 0$:

$$E^y(0,y,t) = 0 \qquad (\text{from } \mathbf{n} \times \mathbf{E} = 0),$$
$$\partial_x E^x(0,y,t) = 0 \qquad (\text{from } \nabla \cdot \mathbf{E} = 0).$$

Taking time derivatives of the above and using the equations:

$$\partial_x^{2m} E^y(0,y,t) = 0 \quad m = 0,1,2,3,\ldots$$
$$\partial_x^{2m+1} E^x(0,y,t) = 0 \quad m = 0,1,2,3,\ldots$$

These *centered* conditions are used on the boundary instead of one-sided approximations.

Centered numerical boundary conditions for high-order approximations

Vector wave equation on a square

$$\mathbf{E}_{tt} = \mathbf{E}_{xx} + \mathbf{E}_{yy} \quad \mathbf{x} \in \Omega = [0,1]^2$$

PEC (perfect electrical conductor) boundary at $x = 0$:

$$\begin{aligned} E^y(0,y,t) &= 0 \qquad (\text{from } \mathbf{n} \times \mathbf{E} = 0), \\ \partial_x E^x(0,y,t) &= 0 \qquad (\text{from } \nabla \cdot \mathbf{E} = 0). \end{aligned}$$

Taking time derivatives of the above and using the equations:

$$\begin{aligned} \partial_x^{2m} E^y(0,y,t) &= 0 \quad m = 0,1,2,3,\ldots \\ \partial_x^{2m+1} E^x(0,y,t) &= 0 \quad m = 0,1,2,3,\ldots \end{aligned}$$

These *centered* conditions are used on the boundary instead of one-sided approximations.

Centered numerical boundary conditions for high-order approximations

Vector wave equation on a square

$$\mathbf{E}_{tt} = \mathbf{E}_{xx} + \mathbf{E}_{yy} \quad \mathbf{x} \in \Omega = [0,1]^2$$

PEC (perfect electrical conductor) boundary at $x = 0$:

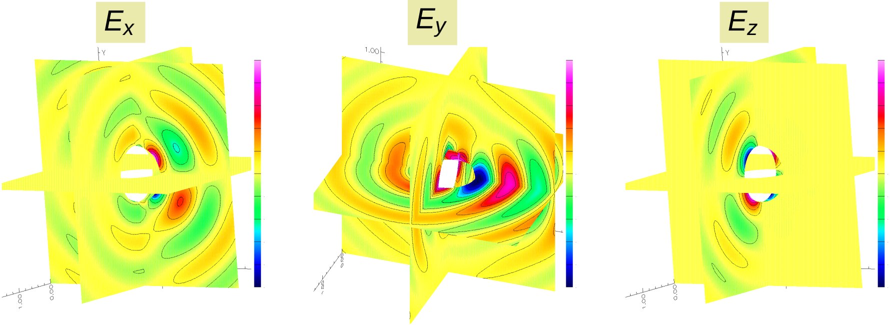
$$\begin{aligned} E^y(0,y,t) &= 0 \qquad \text{(from } \mathbf{n}\times\mathbf{E} = 0\text{)}, \\ \partial_x E^x(0,y,t) &= 0 \qquad \text{(from } \nabla\cdot\mathbf{E} = 0\text{)}. \end{aligned}$$

Taking time derivatives of the above and using the equations:

$$\begin{aligned} \partial_x^{2m} E^y(0,y,t) &= 0 \quad m = 0,1,2,3,\ldots \\ \partial_x^{2m+1} E^x(0,y,t) &= 0 \quad m = 0,1,2,3,\ldots \end{aligned}$$

These *centered* conditions are used on the boundary instead of one-sided approximations.

Scattering of a plane wave by a sphere

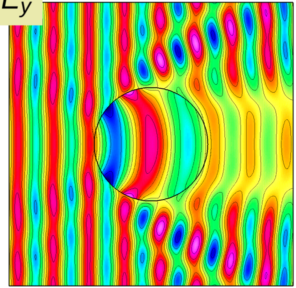


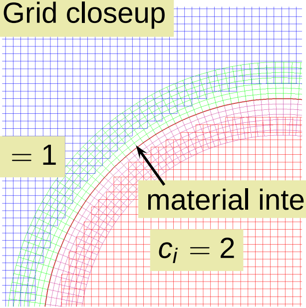
grid	N	$\|\|e^{E_x}\|\|_\infty$	$\|\|e^{E_y}\|\|_\infty$	$\|\|e^{E_z}\|\|_\infty$	$\|\|\nabla \cdot \mathbf{E}\|\|_\infty$
sib1	40	$1.1e-2$	$7.9e-3$	$5.6e-3$	$4.0e-3$
sib2	80	$8.1e-4$	$5.6e-4$	$4.0e-4$	$4.2e-4$
sib4	160	$5.4e-5$	$3.7e-5$	$2.7e-5$	$5.4e-5$
rate		3.84	3.87	3.86	3.10

Maximum errors at $t = 3$.

The finest grid has 6.5 million grid points.

Scattering of a plane wave by a dielectric cylinder

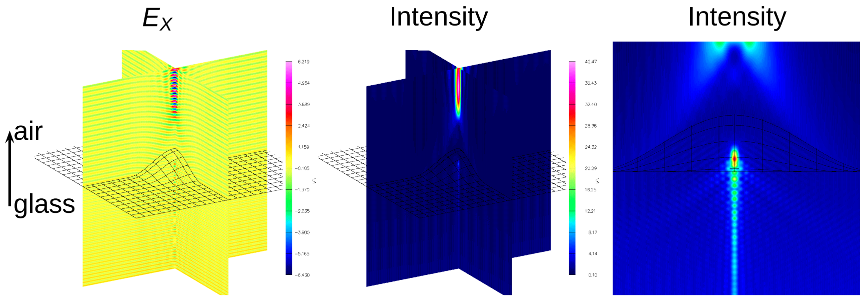




grid	$\|\|e^{E_x}\|\|_\infty$	$\|\|e^{E_y}\|\|_\infty$	$\|\|e^{H_z}\|\|_\infty$	$\delta_{\mathbf{E}}$
$\mathcal{G}_1$	$1.4e{-1}$	$2.9e{-1}$	$3.0e{-1}$	$6.7e{-2}$
$\mathcal{G}_2$	$1.0e{-2}$	$2.1e{-2}$	$2.2e{-2}$	$4.5e{-3}$
$\mathcal{G}_4$	$6.8e{-4}$	$1.4e{-3}$	$1.4e{-3}$	$2.9e{-4}$
rate σ	3.86	3.87	3.88	3.92

Known solution as a Mie series. Maximum errors at $t = 1$.

Scattering by a 3d material interface

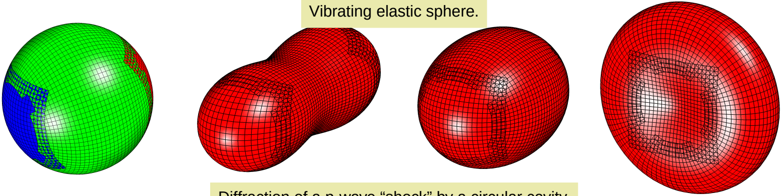


- Uses newly developed 4th-order accurate 3D material interface approximations.
- Scattering of a plane wave by an interface with a bump, glass-to-air.
- 1 billion grid points, 32 nodes (8 processors per node) of a Linux cluster.

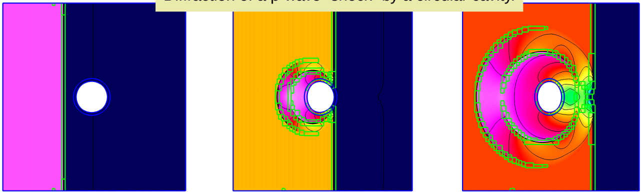
Cgsm: a solid-mechanics solver (in Overture.v24).

- linear elasticity on overlapping grids, with adaptive mesh refinement,
- conservative finite difference scheme for the second-order system,
- upwind Godunov scheme for the first-order-system.

Vibrating elastic sphere.



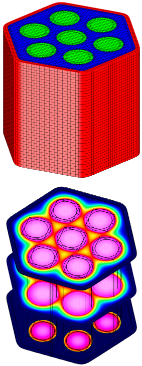
Diffraction of a p-wave “shock” by a circular cavity.



- D. Appelö, J.W. Banks , WDH, D.W. Schwendeman, *Numerical Methods for Solid Mechanics on Overlapping Grids: Linear Elasticity*, LLNL-JRNL-422223, submitted.

Cgmp: a multi-domain multi-physics solver.

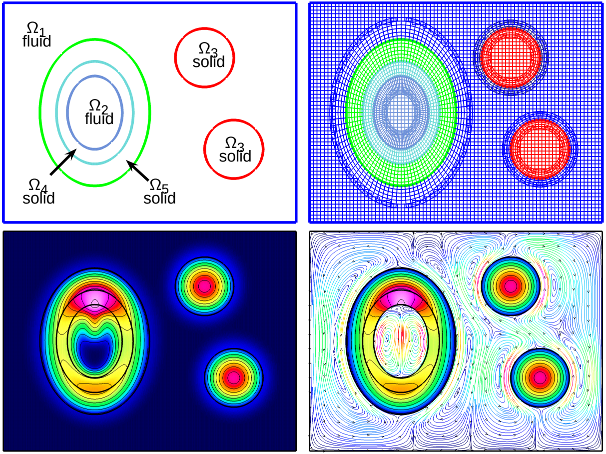
Conjugate heat transfer: coupling incompressible flow to heat conduction in solids.



- overlapping grids for each fluid or solid domain,
- a partitioned solution algorithm (separate physics solvers in each sub-domain),
- (cgins) incompressible Navier-Stokes equations (with Boussinesq approximation) for fluid domains,
- (cgad) heat equation for solid domains,
- a key issue is interface coupling.

• WDH., K. K. Chand, *A Composite Grid Solver for Conjugate Heat Transfer in Fluid-Structure Systems*, J. Comput. Phys, 2009.

The multi-domain composite grid approach for CHT



Each fluid or solid sub-domain is covered by an overlapping grid.
Fluid sub-domains : cgins. Solid sub-domains: cgad.
Coupled problem: cgmp.

Deforming composite grids for FSI

Goal: Couple overlapping grid techniques for modeling fluids and gases (using moving grids and AMR) with linear and non-linear solid mechanics codes.

Approach:

- Fluids: Overlapping grid fluid-mechanics solver.
- Solids : unstructured grid or overlapping-grid solid-mechanics solver.
- Boundary fitted deforming grids are used at the fluid-solid interfaces.

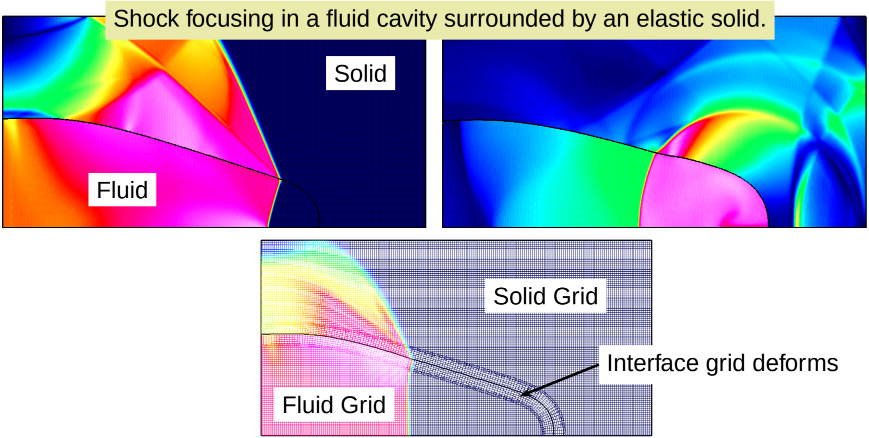
Strengths of the approach:

- Maintains high quality grids for large deformations and displacements.
- Uses efficient structured grid methods optimized for Cartesian grids.

Current status:

- Solve Euler equations in the fluid domains on moving grids.
- Solve equations of linear elasticity in the solid domains.
- Fluid grids at the interface deform over time.

Deforming composite grids for FSI



- Solving the Euler equations in the fluid, linear elasticity in the solid.

The figures show results from preliminary work to model an experiment by Veronica Eliasson.

Conclusions

- Overlapping grids have been used to solve a wide class of problems.
- Smooth boundary fitted grids for accuracy.
- Structured grids for efficiency.
- Rapid grid generation for moving geometry.
- Overture is a toolkit for grid generation and solving PDEs.
- The CG set of PDE solvers solve a variety equation in contiuum mechanics .

Open problem: automatic grid generation for complex geometry.